

AMENDMENTS TO THE CLAIMS

Please cancel claims 24, 26, 30, 34, 38, 39, and 44 without prejudice or disclaimer of their underlying subject matter.

Please amend the claims as follows.

1-22 (Cancelled).

23. (Previously Presented) A parallel processor comprising:

a plurality of processor elements;

a first processor element of said plurality of processor elements for executing a first user program of a plurality of user programs, said first processor element executing a wait instruction, said wait instruction suspending processing of said first user program; and

a second processor element of said plurality of processor elements for executing at least a second user program of said plurality of user programs, said second processor element executing a wait release instruction, said wait release instruction commanding said first processor element to resume said processing of said first user program, said second processor element continuing processing of said second user program after executing said wait release instruction;

a plurality of local memory, each local memory being uniquely associated with a corresponding processor element of said plurality of processor elements; and

a common memory connected to a common bus, said common memory storing said plurality of user programs, a corresponding user program of said plurality of user programs being provided to said corresponding processor element via said common bus, wherein:

said second processor element executes a synchronization wait instruction, said synchronization wait instruction suspending processing of said second user program;

said first processor element executes a program end instruction, said program end instruction resuming said processing of said second user program;

said wait instruction resumes processing of said second user program after said second processor element executes said synchronization wait instruction; and

said first processor element notifies said second processor element that said first processor element is executing said wait instruction; and

said second processor element executes a program execution instruction, said program execution instruction commanding said corresponding processor element to receive said corresponding user program from said common memory and to execute said corresponding user program.

24. (Cancelled)

25. (Previously Presented) A parallel processor as set forth in claim 23, wherein said plurality of processor elements and a common bus for connecting said plurality of processor elements are installed in a single semiconductor chip.

26. (Cancelled)

27. (Previously Presented) A parallel processor as set forth in claim 23, wherein

said plurality of processor elements perform mutually parallel processing on the basis of instructions written in a program; and

said plurality of processor elements are capable of communicating with each other via a common bus.

28. (Cancelled)

29. (Previously Presented) A parallel processor as set forth in claim 23, further comprising another processor element of said plurality of processor elements for executing another user program, said another processor element executing a program end instruction, said program end instruction resuming said processing of said second user program.

30. (Cancelled)

31. (Currently Amended) A parallel processor as set forth in claim 23, ~~30~~, wherein said first processor element is said corresponding processor element and said first user program is said corresponding user program.

32. (Currently Amended) A parallel processor as set forth in claim 23, ~~30~~, wherein said local memory continues to store said user program until said corresponding processor element executes a program end instruction indicating an end of a program.

33. (Currently Amended) A parallel processor as set forth in claim 23, ~~30~~, wherein, when said second processor element enters a waiting state based on said wait instruction, said corresponding processor element which executed said program execution instruction executes said wait release instruction.

34. (Cancelled)

35. (Currently Amended) A parallel processor as set forth in claim 23, ~~34~~, further comprising:

an arbiter for determining which of said plurality of processor elements executes a program instructed to be executed by said program execution instruction, and for reading the program instructed to be executed by said program execution instruction from said common memory to said local memory associated with said corresponding processor element.

36. (Currently Amended) A parallel processing method comprising:

suspending processing of a first user program of a plurality of user programs, said first user program including a wait instruction, a first processor element executing said wait instruction to suspend said processing of said first user program;

resuming said processing of said first user program by executing a wait release instruction, said wait release instruction being included within a second user program of a plurality of user programs, a second processor element of said plurality of processor elements for executing said wait release instruction, said wait release instruction commanding said first processor element to resume said processing of said first user program, said second processor element continuing processing of said second user program after executing said wait release instruction;

suspending processing of said second user program, said second processor element executing a synchronization wait instruction; and

executing a program end instruction to resume said processing of said second user program, ~~wherein~~ said first processor element ~~executing~~ executing said program end instruction, wherein:

said wait instruction resumes processing of said second user program after said second processor element executes said synchronization wait instruction; ~~and~~

said first processor element notifies said second processor element that said first processor element is executing said wait instruction;

a plurality of local memory, each local memory is uniquely associated with a corresponding processor element of said plurality of processor elements;

a common memory connected to a common bus, said common memory storing said plurality of user programs, a corresponding user program of said plurality of user programs being provided to said corresponding processor element via said common bus; and

said second processor element executes a program execution instruction, said program execution instruction commanding said corresponding processor element to receive said corresponding user program from said common memory and to execute said corresponding user program.

37-40. (Cancelled)

41. (Previously Presented) A parallel processing method as set forth in claim 36, wherein, when said second processor element enters a waiting state based on said wait

instruction, said corresponding processor element which executed said program execution instruction executes said wait release instruction.

42. (Previously Presented) A parallel processing method as set forth in claim 36, wherein:

said second processor element executes a program execution instruction, said program execution instruction commanding said corresponding processor element to receive said corresponding user program from said common memory and to execute said corresponding user program.

43. (Currently Amended) A storage medium for storing, in a computer-readable format, the method of claim 36.

~~routines comprising:~~

~~first processing and second processing to be performed in parallel based on instructions written in a plurality of programs, wherein~~

~~said first processing executes a wait instruction in a first program of said plurality of programs, said wait instruction suspending said first processing by entering said first processing into a waiting state;~~

~~said second processing executes a wait release instruction in a second program of said plurality of programs, said wait release instruction resuming execution of said first processing, said second processing continuing execution of said second program after executing said wait release instruction, and~~

~~said second processing enters a synchronization waiting state by executing said wait release instruction until said first processing enters said waiting state when said first processing is not in said waiting state.~~

44. (Cancelled)